VoiceText™

VoiceText Interface SDK for Android

2017-06 Printed in Seoul, Republic of Korea

A open source software was used to develop vtconv API. (APR-iconv library)

OPUS Codec is used for P-type engines.

Table Of Contents

1. Overview

Developed by **Voiceware Co., Ltd.**, **VoiceText** is a solution that reads letters by word. It reads any sentences typed by user, and clearly grasps and analyzes its grammatical structure. It also includes rhyme information of natural words to read with human-like voice.

This product is a high quality and performance speech synthesizer that synthesizes the language of Korean, English, Chinese, Japanese and European languages and etc, using various symbols and is used in many different fields such as communication, finance, public sector and other services.

WIth this solution, it is possible to produce a synthesized audio with high accuracy in pronunciation which sounds impressively natural. It is available on majority of platforms and globally recognized as the world-best synthesizer which runs seamlessly in synthesizing with many voices available.

2. Installation

The environment where the testing done

6 MB to 16 MB memory is used in Android.

- **Device** : Samsung Galaxy Note 4
- **CPU** : Samsung Exynos 5 Octa 5433
- **RAM** : 3GB
- **OS Version** : 6.0.1(Marshmallow)

- **IDE** : Android Studio 3.2.1
- **JRE** : 1.8.0_152
- **JVM** : OpenJDK 64-Bit Server VM By JetBrains s.r.o
- **Min Ver** : Android 4.3(Jelly Bean) - API Level 18

Android VoiceText SDK Package

Android VoiceText SDK consists of jar file and native libraries to use API. Copy it to project to use.

TTS_DB folder

> - TTS DB files

Library folder

> - jniLibs (Native Library)
>   - architectures : `armeabi-v7a`, `arm64-v8a`, `x86`, `x86_64`
>   - libvt_[**language**]_[**type**].so `[type : d16, p16...]`
> - libs (jar) : Enable to use engine API by using native libaray.
>   - avtapi_[info].jar

Doc folder

> - java doc where you can see API information about avtapi_[info].jar

Sample folder

> - Sample project to use SDK API (Android Studio)

How to use SDK

- You can use the synthesis engine API by loading synthesis library. (libvt_[**language**]_[**type**].so `[type : d16, p16...]`)
- The API information is wrapped in JAR lib and API defined in Class CVoiceText is recommended to use.
- You should designate verification.text file or license key and TTS_DB file which is necessary for engine loading. The user can use the API of CVoiceText Class.

Check out the below before using.

> 1. package kr.co.voiceware.comm
>   - Package with IVTDefine.java and MessageTypes.java
>   - IVTDefine.java : Interface to define values needed for development
>   - MessageTypes.java : Message type used in handler
> 2. kr.co.voiceware.voicetext.(samplerate)_(db_size)
>   - Package with VTDemoPlaysound.java
>   - VTDemoPlaysound.java : Sound play demo program of Voice-Text for Android
> 3. **How to verify license**
>
>   There are two ways to verify the license. First is when sample project is given along with **verification.txt** file. The second is when **license key** is provided.

- For verification.txt, : Locate the verification.txt file to the path, “res/raw/”, in project folder. Please refer to the below loadEngine() for more details.
- For license key, : Download the license by using licenseDownload(License-KEY, DB_PATH). Please refer to the below vtLicenseDownload() for more details.

4. **Cautions when verifying license**
   - If wrong license key enters three times when downloading the license (when using licenseDownload()), the IP is shut down so that you are not able to try to download with the same IP for the next 24 hours.
   - Refer to the appendix VwCertificateException
5. TTS DB
   - Save `[tts_single_db.vtdb2]` to an external storage. You can take the path from `Environment.getExternalStorageDirectory().getAbsolutePath()`
   - The external storage is an Android storage. It is the default path when an app being installed to the external storage. It is not what you get as an extra storage by inserting Micro SD card.

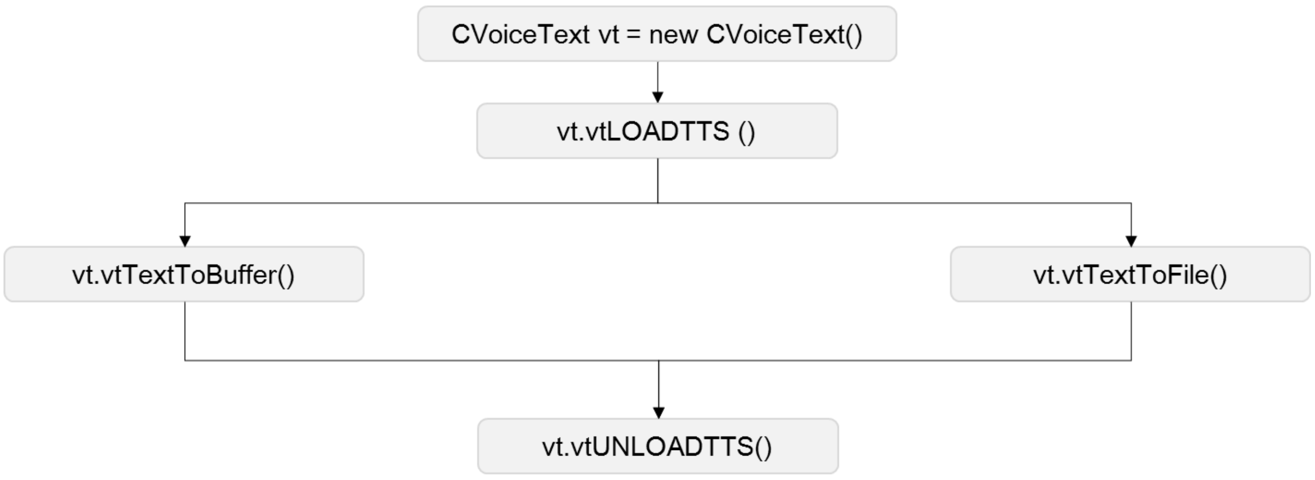
※ With one license key, you can download the verification file multiple times and the Host ID in the Android device is used to check how many verification files are issued. If you use the same Host ID, the number of issuing the file is not increased.

※ Please refer to the information about API regarding API functions, parameters and return values.

3. Getting Started

Work Flows

Flow on using Android VTAPI API



VTAPI Sample Code

Flow on using Android VTAPI API

```
String DbPath = "/sdcard/HoyaSpeech/D16/";
int mSpeakerId = IVTDefine.SPEAKER_ID_JAMES;
```

```
String ttsFileName = DbPath + "tts.pcm";
String ttsText = "Hello Voiceware TTS.";
final byte[] byteLicense = new byte[2048];

CVoiceText vt = new CVoiceText(getApplicationContext(), null);
try {
    getResources().openRawResource(R.raw.verification).read(byteLicense);

    rtnValue = vt.vtLOADTTS(mSelectedTtsDbPath, mSpeakerId, byteLicense);
    rtn = vt.vtTextToFile(mSpeakerId, IVTDefine.VT_FILE_API_FMT_S16PCM, ttsText,
ttsFileName, -1, -1, -1, -1, -1);
    vt.vtUNLOADTTS(mSpeakerId);
} catch (Exception e) {
    Log.e("VOICETEXT", "Failure - " + e.getMessage());
}
```

4. Project Settings

Add library (build.gradle)

```
 dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])
    implementation files('libs/avtapi_[info].jar')
}
```

AndroidManifest.xml [uses-permission]

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="xxx.xxx.xxx">
    <application
        ...
    </application>
    <!-- Permission on using VTAPI and vtlicensemodule.jar(LicenseKey) -->
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <!-- Permission on using vtlicensemodule.jar(LicenseKey) -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
</manifest>
```

Sample Code - Check For Permissions

```
// https://developer.android.com/training/permissions/requesting
@TargetApi(Build.VERSION_CODES.M)
    public void checkForPermissions()
    {
        /**
         * EXTERNAL STORAGE
         */
        if(ContextCompat.checkSelfPermission(this,
Manifest.permission.WRITE_EXTERNAL_STORAGE)
                != PackageManager.PERMISSION_GRANTED
                || ActivityCompat.checkSelfPermission(this,

 Manifest.permission.READ_EXTERNAL_STORAGE)
                != PackageManager.PERMISSION_GRANTED) {
            // Should we show an explanation?
            if(ActivityCompat.shouldShowRequestPermissionRationale(this,
                    Manifest.permission.WRITE_EXTERNAL_STORAGE))
            {
                // Show an expanation to the user *asynchronously* -- don't block
                // this thread waiting for the user's response! After the user
                // sees the explanation, try again to request the permission.
            } else {
                // No explanation needed, we can request the permission.
                ActivityCompat.requestPermissions(this ,
                        new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE,
                                Manifest.permission.READ_EXTERNAL_STORAGE}, 1);
            }
        }


        /**
         * ACCESS, CHANGE WIFI STATE
         */
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION)
                != PackageManager.PERMISSION_GRANTED
                || checkSelfPermission(Manifest.permission.ACCESS_COARSE_LOCATION)
                != PackageManager.PERMISSION_GRANTED) {
            if (ActivityCompat.shouldShowRequestPermissionRationale(this,
                    Manifest.permission.ACCESS_FINE_LOCATION)) {
                // ...
            } else {
                ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.ACCESS_FINE_LOCATION,
                                Manifest.permission.ACCESS_COARSE_LOCATION}, 1);
            }
        }
    }
```

5. Application Programming Interface

Please refer to the attached java_doc folder.

> ## vtLicenseDownload()
>
> ```
> Download license to use Voice-text.
> ```
>
> **Synopsis**
>
> ```
> public class CVoiceText
> public int vtLicenseDownload(String LicenseKEY, String db_path)
> ```
>
> **Parameters**
>
> ```
>  LicenseKEY     License-Key(XXXX-XXXX-XXXX-XXXX)
>  db_path        vtdb Path
> ```
>
> **Description**
>
> ```
> Get license from the server.
> MESSAGE_LICENSE_DOWNLOAD_START comes in to Handler's msg(Message).what when
> starting to download.
> MESSAGE_LICENSE_DOWNLOAD_END comes in to Handler's msg(Message).what when
> finishing the downloading.
> ```
>
> **Return Values**
>
> ```
> [ 0] Success
> [<0] Error.
> ```
>
> **Note** Please display the progress status by using ProgressDialog since it is possible to have some delay due to a slow Internet connection.
>
> There are two ways for license verification.
>
> 1. With verification.txt file
> 2. With license key ※ If you are given the verification.txt file, you don't have to use vtLicenseDownload(). ※ The below three Android permissions are needed if you are to use vtLicenseDownload() after getting license key. “android.permission.INTERNET” “android.permission.ACCESS_WIFI_STATE” “android.permission.CHANGE_WIFI_STATE” "android.permission.ACCESS_NETWORK_STATE" "android.permission.ACCESS_FINE_LOCATION" "android.permission.ACCESS_COARSE_LOCATION"
>
> **Example**
>
> ```
>   CVoiceText vt = new CVoiceText (getApplicationContext(), mHandler);
>   vt.vtLicenseDownload(“xxxx-xxxx-xxxx-xxxx-xxxx”, “/sdcard/HoyaSpeech/D16/”);
> ```

See Also vtLOADTTSEXT() vtIsLicenseExist() vtResetLicens()

vtIsLicenseExist()

```
Check if the license exists.
```

Synopsis

```
public class CVoiceText
public boolean vtIsLicenseExist();
```

Parameters

```
 None
```

Description

```
Check if the license exists through vtLicenseDownload().
```

Return Values

```
true  License exists.
false  License does not exist.
```

Note Only when downloaded license key through licenseDownload().

See Also vtLOADTTSEXT() vtLicenseDownload() vtResetLicens()

vtResetLicense()

```
Use it to download license again.
```

Synopsis

```
public class CVoiceText
public void vtResetLicense();
```

Parameters

```
 None
```

Description

```
Reset flag used in vtIsLicenseExist() in order to download again through
vtLicenseDownload().
```

Return Values

```
None
```

Note Only when downloaded license key through licenseDownload().

See Also vtLOADTTSEXT() vtLicenseDownload() vtIsLicenseExist()

vtLOADTTS()

```
Load synthesizer DB. (verification.txt)
```

Synopsis

```
public class CVoiceText
public int vtLOADTTS(String DB_path, int speakerID, byte[] license)
```

Parameters

```
 DB_path        Synthesis DB path
 speakerID      ID to distinguish speaker (defined in IVTDefine.java)
 license        Verification file
```

Description

```
It is a function to load synthesizer's DB. It is used when the program first
starts.
```

Return Values

```
[ 0] Verifying license is successfully done.
[-1] Verification file is either named wrong or does not exist.
[-2] Data of the file is either damaged or changed or its format is different.
[-3] The validity is expired.
[-4] The Host ID (or Mac address) on the device does not match the infomration in
the license.
[-5] The operating system does not match the information in the license.
[-6] The engine language does not match the information in the license.
[-7] The engine speaker does not match the information in the license.
[-8] The engine version does not match the information in the license.
[-9] The engine DB access type (fire or ram i/o) does not match the information in
the license.
[-10] The engine sampling frequencey does not match the infomration in the
license.
[-11] The engine DB size does not match the information in the license.
[ 1] Attempt to do DB loading with the same engine with different db_path when
using multiple DB.
[ 2] Fail to secure channle memory.
[ 3] Fail to do DB loading related to morphological analysis.
[ 4] Fail to do DB loading related to break index.
[ 5] Fail to do DB loading related to text pre-processing.
[ 6] Fail to do DB loading related to acoustic model.
[ 7] Fail to do DB loading related to selecting unit.
[ 8] Fail to do DB loading related to prosody model.
[ 9] Fail to do audio DB loading.
[10] Fail to do DB loading related to pitch position information.
[11] For other reasons
[75] License is NULL
```

Note Loading should be done before bringing audio buffer.

```
//How to get license
//Locate verification.txt file to (project)/res/raw/ and do the below when doing
vtLOADTTS()
Byte[] byteLicense = new byte[2048];
Int iReadByte =
getResources().openRawResource(R.raw.verification).read(byteLicense);
//Then, put byteLicense into the third parameter of vtLOADTTS().
```

Example

```
Byte[] byteLicense = new byte[2048];
Int iReadByte =
getResources().openRawResource(R.raw.verification).read(byteLicense);
String strDbPath = getResource().getString(R.string.tts_db_path);
CVoiceText vt = new CVoiceText (getApplicationContext(), mHandler);
if(vt.vtLOADTTS(strDbPath, speakerID, byteLicense) =! 0) {
  Log.e("ERROR", "vtLOADTTS() ERROR");
  return;
}
```

See Also

vtLOADTTSEXT()

```
Load synthesizer's DB. (License kEY)
```

Synopsis

```
public class CVoiceText
public int vtLOADTTSEXT(String DB_path, int speakerID, String licensePath)
```

Parameters

```
 DB_path        Synthesis DB path
 speakerID      ID to distinguish speaker (defined in IVTDefine.java)
 licensePath    License path
```

Description

```
It is a function to load synthesizer's DB. It is used when the program first starts.
```

Return Values

```
[ 0] Verifying license is successfully done.
[-1] license does not exist.
[-2] Invalid format (License either changed or wrong.)
[-3] License validity is expired.
[-4] Either Host ID does not exist or different one.
[-5] license does not exist.
[ 1] Attempt to do DB loading with the same engine with different db_path when
using multiple DB.
[ 2] Fail to secure channle memory.
[ 3] Fail to do DB loading related to morphological analysis.
[ 4] Fail to do DB loading related to break index.
[ 5] Fail to do DB loading related to text pre-processing.
[ 6] Fail to do DB loading related to acoustic model.
[ 7] Fail to do DB loading related to selecting unit.
[ 8] Fail to do DB loading related to prosody model.
[ 9] Fail to do audio DB loading.
[10] Fail to do DB loading related to pitch position information.
[11] For other reasons
[75] License is NULL
```

Note It should be loaded before getting audio buffer or playing. You should get MESSAGE_LICENSE_DOWNLOAD_END with msg(Message).what of Handler and then call loadEngineExt(). Download license through vtLicenseDownload(“License-KEY”, “DB_PATH”). **Example**

```
String strDbPath = getResource().getString(R.string.tts_db_path);
CVoiceText vt = new CVoiceText (getApplicationContext(), mHandler);
if(vt.vtLOADTTSEXT(strDbPath, speakerID, "/sdcard/HoyaSpeech/D16/")) {
  Log.e("ERROR", "vtLOADTTSEXT() ERROR");
  return;
}
```

See Also vtIsLicenseExist() vtResetLicense() vtLicenseDownload()

vtUNLOADTTS()

```
Unload synthesizer's DB.
```

Synopsis

```
public class CVoiceText
public void vtUNLOADTTS(int speakerID);
```

Parameters

```
 speakerID      ID to distinguish speaker (defined in IVTDefine.java)
```

Description

```
It is a function to free allocated memory by unloading synthesizer's DB. It is used
when closing the program.
```

Return Values

```
None
```

Note

Example

```
```

See Also vtIsLicenseExist() vtLOADTTS()

vtLOADUserDict()

```
Load user dictionary.
```

Synopsis

```
public class CVoiceText
public int vtLOADUserDict(int speakerID, int dictidx, String fileName);
```

Parameters

```
  speakerID      ID to distinguish speaker (defined in IVTDefine.java)
  dictidx        Index to distinguish when multiple user dictionaries used
  fileName       User dictionary name (including
 path,ex."/sdcard/userdict_eng.csv")
```

Description

```
It is a function to load another user dictionary not included in synthesis DB.
It should be used after loading synthesis DB done and it is used when synthesizing
dictidx.
```

Return Values

```
[ 1] Loading successfully done.
[-1] dictidx has a value out of valid range.
[-2] There is a user dictionary whose dictidx is already loaded.
[-3] Either no user dictionary or valid entry so that loading fails.
[-4] For other reasons
```

Note In Android, only one user dictionary is recognized in addition to the default user dictionary. In other words, dictidx is one. **Example**

```
if(vt.vtLOADUserDict(speakerID, 1, “/sdcard/userdict_eng.csv”) != 1)
{
  Log.e("ERROR", "vtLOADUserDict() ERROR");
  return;
}
```

See Also vtUNLOADUserDict()

vtUNLOADUserDict()

```
Unload user dictionary.
```

Synopsis

```
public class CVoiceText
public int vtUNLOADUserDict(int speakerID, int dictidx);
```

Parameters

```
  speakerID      ID to distinguish speaker (defined in IVTDefine.java)
  dictidx        Index to distinguish when multiple user dictionaries used
```

Description

```
It is a function to unload a user dictionary.
If a user dictionary is unloaded, it is automatically unloaded in unloading
synthesis DB.
```

Return Values

```
[ 1] Unloading successfully done.
[-1] User dictionary which dictidx points is already unloaded.
[-2] dictidx has a value out of valid range.
[-3] For other reasons
```

Note

Example

```
if(vt.vtUNLOADUserDict(speakerID, 1) != 1)
{
  Log.e("ERROR", "vtUNLOADUserDict() ERROR");
  return;
}
```

See Also vtUNLOADUserDict()

vtTextToFile()

```
Save the outout into file
```

Synopsis

```
public class CVoiceText
public int vtTextToFile( int speakerID, int fmt, String text, String fileName, int
pitch, int speed, int volume, int pause int dictidx);
```

Parameters

```
 speakerID      ID to distinguish speaker (defined in IVTDefine.java)
 fmt            Define the format of output (defined in IVTDefine.java)
 VT_FILE_API_FMT_S16PCM         16bits Linear PCM
 VT_FILE_API_FMT_ALAW       8bits A-law PCM
 VT_FILE_API_FMT_MULAW      8bits Mu-law PCM
 VT_FILE_API_FMT_DADPCM         4bits Dialogic ADPCM
 VT_FILE_API_FMT_S16PCM_WAVE    16bits Linear PCM WAVE
 VT_FILE_API_FMT_U08PCM_WAVE    8bits Unsigned Linear PCM WAVE
 VT_FILE_API_FMT_ALAW_WAVE      8bits A-law PCM WAVE
 VT_FILE_API_FMT_MULAW_WAVE     8bits Mu-law PCM WAVE
 VT_FILE_API_FMT_MULAW_AU   8bits Mu-law PCM SUN AU
 text           Text to synthesize
 fileName       Filename to save the output into with save path included.
 pitch          pitch set to output. The initial value is set with 100(%).
                The range is 50 to 200(%) and the initial value is used when it is
-1.
 speed          Speed set to output. The initial value is set with 100(%).
                The range is 50 to 400(%) and the initial value is used when it is
-1.
 volume         Volume set to output. The initial value is set with 100(%).
                The range is 0 to 500(%) and the initial value is used when it is
-1.
 pause          Pause set to output. The initial value is set to 687 (msec).
                The range is 0 to 65535 (msec) and the initial value is used when
it is -1.
 dictidx        Index to distinguish when multiple user dictionaries used.
```

Description

```
Synthesis DB should be loaded before this function is used.
```

Return Values

```
[ 1] Successfully saved the outout into file
[-1] Used unsupported format for the output
[-2] Fail to secure channel memory
[-3] Text characters are NULL pointer.
[-4] The length of text is 0.
[-5] Synthesis DB is not loaded.
[-6] Fail to create output file
[-7] For other reasons
```

Note

Example

```
if(vt.vtTextToFile(VT_FILE_API_FMT_S16PCM_WAVE ,speakerID, “hello”,
“/sdcard/hello.wav”, -1,-1,-1,-1, -1) != 1)
{
  Log.e("ERROR", "vtTextToFile() ERROR");
  return;
}
```

See Also

vtTextToBuffer()

```
Save the output to PCM frame buffer
```

Synopsis

```
public class CVoiceText
public byte[] vtTextToBuffer( int speakerID, int fmt, String text,, int flag
String fileName, int pitch, int speed, int volume, int pause int dictidx);
```

Parameters

```
 speakerID      ID to distinguish speaker (defined in IVTDefine.java)
 fmt            Define the format of output (defined in IVTDefine.java)
 VT_BUFFER_API_FMT_S16PCM       16bits Linear PCM
 VT_BUFFER_API_FMT_ALAW         8bits A-law PCM
 VT_BUFFER_API_FMT_MULAW        8bits Mu-law PCM
 VT_BUFFER_API_FMT_DADPCM       4bits Dialogic ADPCM
 text           Text to synthesize
 flag           Flag to define the first move of API
                When getting the first frame, it is 0.
```

```
                  It is 1 from the second frame.
                  Use 2 when cancelling the synthesis in the middle of process and
throwing the rest frames.
 pitch            pitch set to output. The initial value is set with 100(%).
                  The range is 50 to 200(%) and the initial value is used when it is
-1.
 speed            Speed set to output. The initial value is set with 100(%).
                  The range is 50 to 400(%) and the initial value is used when it is
-1.
 volume           Volume set to output. The initial value is set with 100(%).
                  The range is 0 to 500(%) and the initial value is used when it is
-1.
 pause            Pause set to output. The initial value is set to 687 (msec).
                  The range is 0 to 65535 (msec) and the initial value is used when
it is -1.
 dictidx          Index to distinguish when multiple user dictionaries used.
```

Description

```
Synthesis DB should be loaded before this function is used.
```

Return Values

```
[null] fail
*Get a return value by using TextToBufferRTN() to see if the synthesis is done
successfully.
When it is successful, it returns 0 if it is the frame except for the last one while
it returns 1 if it is the last frame.
[ 1] Successfully saved the outout into file
[-1] Used unsupported format for the output
[-2] Fail to secure channel memory
[-3] Text characters are NULL pointer.
[-4] The length of text is 0.
[-5] Synthesis DB is not loaded.
[-6] Fail to create output file
[-7] For other reasons
```

Note The frame size is 60 KByte to support. If the entire size is 60 KByte or more than it, the first frame is 60 KByte and the last frame is 60 KByte or less than it. Therefore, the synthesis is finished if the returned byte[] is 60 KByte or less than it.

Example

```
//Example showing how to play with buffer by using audiotrack
byte[] testBuff=null;
AudioTrack testAudio;
List<Byte> read_total = new ArrayList<Byte>();
int testFlag=0;
int counter=0;
while(true) {
  testBuff = vt.textToBuffer(IVTDefine.TTS_BRIDGET_DB, "hello, this is the
voiceware Demo. test", testFlag);
```

```
    //TextToBufferRTN
    Log.e(“example”, "engine returns " +
  vt.TextToBufferRTN(IVTDefine.TTS_BRIDGET_DB));
    counter=counter+testBuff.length;
    for (int iT = 0; iT < testBuff.length; iT++)
    {
        read_total.add(testBuff[iT]);
    }
    if(testBuff.length!=60000) break;
  }
  byte[] total_buffer = new byte[read_total.size()];
  for (int iT = 0; iT < read_total.size(); iT++) {
    total_buffer[iT] = read_total.get(iT);
  }
  testAudio = new AudioTrack(AudioManager.STREAM_MUSIC, 16000,
  AudioFormat.CHANNEL_CONFIGURATION_MONO, AudioFormat.ENCODING_PCM_16BIT,
  total_buffer.length, AudioTrack.MODE_STATIC);
  testAudio.write(total_buffer, 0, total_buffer.length);
  testAudio.play();
```

See Also

vtSetPitchSpeedVolumePause()

```
Control the output in pitch, speed, volume and pauses between sentences.
```

Synopsis

```
public class CVoiceText
public void vtSetPitchSpeedVolumePause(int speakerID, int pitch, int speed, int
volume, int pause);
```

Parameters

```
  speakerID     ID to distinguish speaker (defined in IVTDefine.java)
  pitch         pitch set to output. The initial value is set with 100(%).
                The range is 50 to 200(%) and the initial value is used when it is
-1.
  speed         Speed set to output. The initial value is set with 100(%).
                The range is 50 to 400(%) and the initial value is used when it is
-1.
  volume        Volume set to output. The initial value is set with 100(%).
                The range is 0 to 500(%) and the initial value is used when it is
-1.
  pause         Pause set to output. The initial value is set to 687 (msec).
                The range is 0 to 65535 (msec) and the initial value is used when
it is -1.
```

Description

```
Load synthesis DB. Then, set up before synthesizing audio.
Find maximum and minimum values through vtGetTTSInfo().
```

Return Values

```
None
```

Note

Example

```

```

See Also vtGetTTSInfo()

vtSetCommaPause()

```
Control the output in pitch, speed, volume and pauses between sentences.
```

Synopsis

```
public class CVoiceText
public void setCommaPause(int speakerID, int pause);
```

Parameters

```
 speakerID      ID to distinguish speaker (defined in IVTDefine.java)
 pause          Set comma pause in the output. The initial value is set to 200
(msec).

                The range is 0 to 65535 (msec) and the initial value is used when
it is -1.
```

Description

```
Load synthesis DB. Then, set up before synthesizing audio.
Find maximum and minimum values through vtGetTTSInfo().
```

Return Values

```
None
```

Note

Example

```

```

See Also vtGetTTSInfo()

vtGetTTSInfo()

```
Get information about VoiceText engine.
```

Synopsis

```
public class CVoiceText
public String vtGetTTSInfo(int speakerID, int request);
```

Parameters

```
 speakerID      ID to distinguish speaker (defined in IVTDefine.java)
 request        Select information to get from VoiceText engine. (defined in
IVTDefine.java)
        VT_TTS_INFO_BUILD_DATE      Date on engine compiled
        VT_TTS_INFO_VERIFY_CODE     Verification result : success (0)
        VT_TTS_INFO_MAX_CHANNEL     The maximum number of channels in verification
file
        VT_TTS_INFO_DB_DIRECTORY    Designated path when synthesis DB installed
        VT_TTS_INFO_LOAD_SUCCESS_CODE   Return code when synthesis DB loading done
successfully
        VT_TTS_INFO_MAX_SPEAKER     The maximum number of speakers engine supports
        VT_TTS_INFO_DEF_SPEAKER     Basic speaker ID set to engine
        VT_TTS_INFO_DB_ACCESS_MODE  Synthesis DB input/output type : file (0), ram
(1)
        VT_TTS_INFO_SAMPLING_FREQUENCY Sampling frequency for output
        VT_TTS_INFO_MAX_PITCH_RATE  The maximum pitch set to engine
        VT_TTS_INFO_DEF_PITCH_RATE  The default pitch set to engine
        VT_TTS_INFO_MIN_PITCH_RATE  The minimum pitch set to engine
        VT_TTS_INFO_MAX_SPEED_RATE  The maximum speed set to engine
        VT_TTS_INFO_DEF_SPEED_RATE  The default speed set to engine
        VT_TTS_INFO_MIN_SPEED_RATE  The minimum speed set to engine
        VT_TTS_INFO_MAX_VOLUME      The maximum volume set to engine
        VT_TTS_INFO_DEF_VOLUME      The default volume set to engine
        VT_TTS_INFO_MIN_VOLUME      The minimum volume set to engine
        VT_TTS_INFO_MAX_SENT_PAUSE  The pause between the maximum sentences set to
engine
        VT_TTS_INFO_DEF_SENT_PAUSE  The pause between the default sentences set to
engine
        VT_TTS_INFO_MIN_SENT_PAUSE  The pause between the minimum sentences set to
engine
        VT_TTS_INFO_DB_BUILD_DATE   Date on DB being created from Embedded engine
        VT_TTS_INFO_MAX_COMMA_PAUSE The maximum comma pause set to engine
        VT_TTS_INFO_DEF_COMMA_PAUSE The default comma pause set to engine
        VT_TTS_INFO_MIN_COMMA_PAUSE The minimum comma pause set to engine
```

Description

```
It is possible to use even before synthesis DB is loaded.
It is possible to get much information about engine so that it may be used to check
an error when building an application.
```

Return Values

```
information in the form of string
```

Note

Example

```

```

See Also

vtSetParenthesisCharNumber()

```
Control on reading text characters surrounded with parenthesis.
```

Synopsis

```
public class CVoiceText
public void vtSetParenthesisCharNumber(int speakerID, int nByte);
```

Parameters

```
 speakerID      ID to distinguish speaker (defined in IVTDefine.java)
 nByte          Define reading rule on text characters surrounded with
parenthesis.
                It reads if the length is nByte or longer while it skips the
reading if it is lower than nByte.
                If nByte is 0, the text characters are skipped.
```

Description

```
It is possible to use even before synthesis DB is loaded.
```

Return Values

```
None
```

Note

Example

See Also

6. Appendix A : List on Errors

VwCertificateException

```
public static final int  ERROR_CODE_LICE_M2 = -2;
public static final String  ERROR_MSG_LICE_M2 = "Invalid CD-KEY.";
public static final int  ERROR_CODE_LICE_M3 = -3;
public static final String  ERROR_MSG_LICE_M3 = "CD-KEY : Issued maximum count has been
exceeded.";
public static final int  ERROR_CODE_LICE_M4 = -4;
public static final String  ERROR_MSG_LICE_M4 = "That do not have access CD-KEY";
public static final int  ERROR_CODE_LICE_M100 = -100;
public static final String  ERROR_MSG_LICE_M100 = "If the cd-key is requested over than
3times. The IP address that has been requesting the cd-key is blocked by the server.";
public static final int  ERROR_CODE_LICE_M300 = -300;
public static final String  ERROR_MSG_LICE_M300 = "HostID is incorrect.";
public static final int  ERROR_CODE_LICE_M55 = -55;
public static final String  ERROR_MSG_LICE_M55 = "It occurres UNKNOWN EXCEPTION.";
public static final int  ERROR_CODE_LICE_M56 = -56;
public static final String  ERROR_MSG_LICE_M56 = "It has no Certificate's contents.";
public static final int  ERROR_CODE_LICE_M57 = -57;
public static final String  ERROR_MSG_LICE_M57 = "Invalid format for License";
public static final int  ERROR_CODE_LICE_M101 = -101;
public static final String  ERROR_MSG_LICE_M101 = "Android context path is NULL.";
public static final int  ERROR_CODE_LICE_M102 = -102;
public static final String  ERROR_MSG_LICE_M102 = "It has no Device's info file.";
public static final int  ERROR_CODE_LICE_M103 = -103;
public static final String  ERROR_MSG_LICE_M103 = "It has no Certificate's file";
public static final int  ERROR_CODE_LICE_M104 = -104;
public static final String  ERROR_MSG_LICE_M104 = "It has no Certificate's contents.";
public static final int  ERROR_CODE_LICE_M105 = -105;
public static final String  ERROR_MSG_LICE_M105 = "WifiService: Current process has
android.permission.ACCESS_WIFI_STATE. , android.permission.CHANGE_WIFI_STATE";
public static final int  ERROR_CODE_LICE_M106 = -106;
public static final String  ERROR_MSG_LICE_M106 = "Wifi' MacAddress is NULL.";
public static final int  ERROR_CODE_LICE_M107 = -107;
public static final String  ERROR_MSG_LICE_M107 = "INTERNET: Current process
hasandroid.permission.INTERNET. or Is not connected to the Internet. ";
public static final int  ERROR_CODE_LICE_M108 = -108;
public static final String  ERROR_MSG_LICE_M108 = "File: Current process has
android.permission.WRITE_EXTERNAL_STORAGE.";
public static final int  ERROR_CODE_COMM_155 = -155;
public static final String  ERROR_MSG_COMM_155 = "It occurres UNKNOWN EXCEPTION.";
public static final int  ERROR_CODE_COMM_156 = -156;
public static final String  ERROR_MSG_COMM_156 = "Parameter is NULL.";
```